Intro

Welcome to the Art of Product Engineering! The lab portion for this class will focus on building a simple IoT device using a Raspberry Pi connected to a custom PCB with an array of sensors. You will begin with a minimal setup, with just the Raspberry Pi and a skeleton 'main' file. Every lab session will build up on the previous week's work, so it is your responsibility to be up to date with all the material.

Your lab objectives for this session are:

- Assembly of the development environment.
- Demonstration of basic Software Engineering concepts, in Python.
- Comfortability with the tools necessary for the course.

Raspberry Pi is a simple single-board computer that makes prototyping accessible to all levels of experience, giving electronic hobbyists a cheap and easy entry point for projects. We will be using the Raspberry Pi for this class because of its educational value. Raspberry Pi runs the Raspbian operating system by default, which is a custom version of Debian Linux distro. Knowledge of the command line is paramount for success in developing for this OS.

Python is a scripting language with a high level of flexibility and relative ease of use. Raspbian and Python play very well together, easily allowing users to do a wide variety of tasks, from complex mathematical operations, to accessing the GPIO pins of a Pi and interfacing with the real world.

In order to succeed in this course, you must be comfortable and familiar with Python and the command line. Here are some tutorials to help you with these tools:

- https://www.codecademy.com/ : Offers tutorials for both the command line and Python. Completing these should provide more than enough familiarity for this course.
- http://linuxcommand.org/lc3_learning_the_shell.php : Offers and incredible in-depth tutorial, beyond the scope of this class.
- https://learnpythonthehardway.org/book/ : Extensive overview of the Python language.

You will not be tested on the content of these tutorials, but if you are unable to complete them, you will be unable to achieve anything in this course.

Raspberry Pi Setup

The first step necessary to working on the Pi is to install the operating system, Raspbian. Since this installation process generally requires a monitor, keyboard, and mouse (which we have none), we went ahead and performed that step for you.

If you are curious about how this process goes, please visit:

https://www.raspberrypi.org/documentation/setup/

Your baseline setup includes just the Pi with Raspbian install. But as we previously mentioned, you do not necessarily have access to a monitor, keyboard, or mouse, so how can you possibly work with this device? There are many options to accomplish this task, and we will discuss several of them. Most tools require a functioning internet connection, but in the current state, the Pi is not connected to any network. This fact leaves us with one option: the Console Cable.

The console cable allows the user to communicate with the Pi through another computer. It connects to the Pi's GPIO pins to the host computer's USB port. This process requires a bit of set up, which is extensively documented here:

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable?view=all

Depending on what operating system your host computer uses, there will be different ways of setting up the console cable. Make sure to follow their instructions carefully.

Note: You don't need to worry about enabling UART on boot config, that has already been done for you.

The console cable's main advantage is that it allows the user to communicate with the Pi even when there is no network available. Yet one big disadvantage is the fact that this method is much slower than most other options, so we recommend using this only when a network connection (SSH, VNC) is not available.

WARNING: As stated in the Adafruit tutorial, connecting the 5V (red wire) to the Pi's GPIO while it is also being powered by the micro-USB cable can be dangerous. Use one or the other, but certainly not both.

Network Setup

Now that you are able to communicate with the Pi, it's time to start configuring its options. First and foremost, let us connect our Pi to the Internet. With the console cable, we have access to the terminal, so make sure you are comfortable using the command line. If not, follow the tutorials at the beginning of this guide.

For our Pi to connect to a WiFi access point, two configuration files must be modified. In order to modify these files, we will be using the Raspberry Pi's default text editor, nano. nano is a very simple terminal-based text editor, which will be enough for network configuration, but is not powerful enough for the remainder of the class (more on that later). In order to use nano, simply type:

```
nano path/to/file
```

But since the files we are modifying are protected, you must have root privileges. This is done with:

```
sudo nano path/to/file
```

Now we must modify the files, which are located at:

`/etc/network/interfaces`
And
`/etc/wpa_supplicant/wpa_supplicant.conf`

Interfaces setup

In order to connect to UCSD's complicated Enterprise network, modify your `/etc/network/interfaces` file so it looks like:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:

source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet dhcp
        pre-up wpa_supplicant -B -Dwext -i wlan0
-c/etc/wpa_supplicant/wpa_supplicant.conf
        post-down killall -q wpa_supplicant

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Note: Text in red is in a single line. It is too long so it "wrapped around".

WPA config

Different network types require different configurations. The ones you are most likely to use are WPA Personal and WPA Enterprise. WPA Enterpirse is used for UCSD-PROTECTED and WPA Personal is commonly used in home routers (you probably use this at home) In order to add new networks, ***add*** the example code to `/etc/wpa_supplica nt/wpa_supplicant.conf`

Enterprise

```
ctrl_interface=/var/run/wpa_supplicant
network={
      ssid="UCSD-PROTECTED"
      scan_ssid=1
      key_mgmt=WPA-EAP
      pairwise=CCMP TKIP
      group=CCMP TKIP
      eap=PEAP
      identity="UCSD_AD_USERNAME"
      password="UCSD_AD_PASSWORD"
      phase1="peapver=0"
      phase2="MSCHAPV2"
}
```

Replace `UCSD_AD_USERNAME` and `UCSD_AD_PASSWORD` with your UCSD username and password.

Home network

```
network={
        ssid="HOME_NETWORK_SSID"
        psk="HOME_NETWORK_PASSWORD"
        key_mgmt=WPA-PSK
}
```

Replace ssid and psk accordingly.

In order to add more networks, simply append more of the `network={...}` blocks. If you wish to learn more about wpa_supplicant, follow https://linux.die.net/man/8/wpa_supplicant

After you have configured your Pi to connect to WiFi, reboot the system by running `sudo reboot` and wait until you are able to log back in. Once back in the terminal, run:

```
ping -c 3 8.8.8.8
```

This will ping google's servers 3 times! If it does not work, your network might be down or you might have incorrectly configured your network.

In order to obtain your Pi's local IP address (which will come in handy very soon!) run:

```
hostname -I
```

Tools of the trade

We will be using a wide variety of tools for this class, including SSH, VNC, git, and vim. Generally speaking, when installing some piece of software that is officially hosted by the distribution, run:

```
sudo apt-get install [package-name]
```

So let's try this out with vim! Run:

```
sudo apt-get install vim
```

And follow the on screen instructions to finalize the installation.

Vim is a very powerful text editor which is widely used in the industry, an indispensable tool for your career. Just like nano, it is terminal-based, but it is significantly more powerful. Follow http://www.openvim.com/ for a thorough tutorial on how to use this piece of software. From now on, we will be using vim for everything.

SSH Setup

SSH (Secure Shell) is a network protocol that allows users to log into remote computer systems. Although the Console Cable already provides us the ability to communicate with the Pi, SSH is faster, more powerful, and requires no additional hardware. We recommend using SSH as the main communication protocol, reserving console cable use to situations where the Pi is unable to connect to a network.

In order to set up SSH, follow Adafruit's tutorial:
https://learn.adafruit.com/adafruits-raspberry-pi-lesson-6-using-ssh?view=all

Once SSH is setup, all you need to know is you Pi's IPv4 address, which can be found by running the command:

```
hostname -I
```

VNC Setup

VNC (Virtual Network Computing) is a protocol that allows sharing of the desktop graphical environment. What this means is users gain that ability to operate a remote computer using the computer's graphical interface. You can think of it as SSH for GUI. The downside to using VNC is that it requires a much larger bandwidth, thus making the process significantly slower than SSH. We recommend using VNC only when dealing with the Pi camera, and using SSH for everything else.

In order to set up VNC, follow Adafruit's tutorial:
https://learn.adafruit.com/adafruit-raspberry-pi-lesson-7-remote-control-with-vnc/overview

Git Setup

Version control is a very important aspect of software development. Keeping track of all the changes in a codebase is useful in many situations, such as error tracking, code sharing, and resolving conflicts. Git is the most popular and robust Distributed Version Control System (DVCS). We will be using GitHub for hosting and grading all of our code. Go to www.github.com and create an account.

An introductory tutorial to GitHub can be found at:
https://guides.github.com/activities/hello-world/
Once you understand the purpose of git, follow the command line tutorial as well:
https://try.github.io/levels/1/challenges/1

In order to install git on your pi, run

```
sudo apt-get install git
```

If you want to learn pretty much everything there is to git, you can get their free ebook at:
https://git-scm.com/book/en/v2
(Reading just the first 3 chapters will give you great insight on the software!)

A cheatsheet can also be found at:
https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf

Task 1:

Create a GitHub repo for your group. Name it:

"SmartPlug-USERNAME1-USERNAME2-USERNAME3-...USERNAMEN"

Replacing USERNAME1-N with your team member's UCSD usernames, in alphabetical order. For example, Raul and Wilson's UCSD emails are rpegan@eng.ucsd.edu and witran@ucsd.edu, so the repo would be called:

"SmartPlug-rpegan-witran"

Turnin Instructions

Lab 1 does not require much of a lab write-up, since it only covers setting up your development environment.

Email one pdf document per group to your TA, and include the following information:

- Your group's GitHub repo URL.
- A short description of your plans and ideas for the SmartPlug.

Be sure to include all of the team member's names and PIDs on top of the document.